

Contents

1	Final Métodos Numéricos	1
1.1	Eliminación Gaussiana	1
1.2	Factorización LU	1
1.3	Matrices SDP	3
1.4	Factorización QR	4
1.5	Autovalores	6
1.6	Factorización SVD	7
1.7	Métodos Iterativos	8
1.8	Cuadrados Mínimos	9
1.9	Interpolación	10
1.10	Aritmética de la computadora	13

1 Final Métodos Numéricos

1.1 Eliminación Gaussiana

Motivación: Resolver sistemas de ecuaciones matriciales de la forma $Ax = b$.

Algoritmo *Eliminación Gaussiana*, complejidad $\mathcal{O}(n^3)$. La idea es ir poniendo 0 en las columnas hasta triangular la matriz. Condición necesaria: $a_{ii}^{i-1} \neq 0 \forall i = 1 \dots n$

Pero se puede arreglar en caso de que $a_{ji}^{i-1} = 0 \forall j = i + 1 \dots n$ ya que podemos saltar este paso. Si existe $a_{ji}^{i-1} \neq 0$, permutamos las filas y seguimos.

- **Pivoteo parcial:** En cada paso utilizar el a_{ji} con mayor valor absoluto para minimizar el error numérico (dividir por número más grande).
- **Pivoteo completo:** Lo mismo que parcial, pero buscando el mayor también en la fila. Es más costoso y requiere memorizar permutación de columnas.

Utiliza fuertemente que dos sistemas de ecuaciones son equivalentes \iff 1. Sumar/restar ecuaciones 2. Multiplicar ecuaciones por un escalar no nulo 3. Permutar ecuaciones

Utilizando esto, se busca un sistema equivalente pero donde la matriz sea triangular superior, luego utilizando *backward substitution* se encuentra la solución reemplazando.

- Si la matriz ya está triangulada $\implies \mathcal{O}(n^2)$
- Si la matriz es diagonal $\implies \mathcal{O}(n)$

1.2 Factorización LU

Motivación: Resolver sistemas de ecuaciones de manera más eficiente. Reducir el costo de $\mathcal{O}(n^3)$ a $\mathcal{O}(n^2)$

Para resolver $Ax = b$, si conozco L triangular inferior, U triangular superior tal que $A = LU$, entonces puedo resolver:

$$Ax = b \iff LUx = b \iff Ux = y \wedge Ly = b$$

Resolviendo así dos sistema de orden $\mathcal{O}(n^2)$.

- No toda matriz tiene factorización LU .

La factorización se encuentra utilizando el algoritmo de *Eliminación Gaussiana*, escribiendo los pasos de triangulación como ecuaciones matriciales:

$$M^{n-1}M^{n-2} \dots M^1 A = U$$

donde M^i es la matriz que al multiplicar a A a izquierda pone 0 en la columna i .

$$M^1 = \begin{pmatrix} 1 & 0 & 0 \\ -m_{21} & 1 & 0 \\ -m_{31} & 0 & 1 \end{pmatrix}, \text{ con } m_{ij} = \frac{a_{ij}}{a_{ii}}$$

Si el algoritmo de EG no funciona ($a_{ii} = 0$), no podremos encontrar estas matrices. Además las matrices M^i son triangulares inferiores e inversibles (y la inversa es triangular inferior) $\Rightarrow M^{n-1} \dots M^1 = \hat{L}$ es triangular inferior e inversible.

$$\hat{L}A = U \Leftrightarrow A = \hat{L}^{-1}U \Leftrightarrow A = LU$$

Propiedades:

- En esta factorización LU se cumple: $l_{ii} = 1 \forall i$
- Si A es inversible y tiene LU , entonces es única
- Si A tiene todas sus submatrices principales inversibles, entonces tiene factorización LU
- Si A es *edd*, entonces tiene factorización LU

Toda matriz A tiene factorización PLU , que proviene del algoritmo de *Eliminación Gaussiana* al permutar filas para evitar $a_{ii} = 0$. En este caso se obtiene $PA = LU$, donde P es la permutación de filas realizadas al triangular.

- Toda matriz tiene factorización PLU .

$$Ax = b \Leftrightarrow PAx = Pb \Leftrightarrow LUx = Pb \Leftrightarrow Ux = y \wedge Ly = Pb$$

Y calcular Pb es “fácil” por ser una matriz de permutación. ## Normas y número de condición

$f(x) : \mathbb{R}^n \rightarrow \mathbb{R}$ es una norma vectorial \Leftrightarrow

1. $f(x) > 0$ si $x \neq 0$
2. $f(x) = 0 \Leftrightarrow x = 0$
3. $f(\alpha x) = |\alpha|f(x) \forall \alpha \in \mathbb{R}$
4. $f(x + y) \leq f(x) + f(y)$

$F(A) : \mathbb{R}^{m \times n} \rightarrow \mathbb{R}$ es una norma matricial \Leftrightarrow

1. $F(A) > 0$ si $A \neq 0$
2. $F(A) = 0 \Leftrightarrow A = 0$

3. $F(\alpha A) = |\alpha|F(A) \forall \alpha \in \mathbb{R}$
4. $F(AB) \leq F(A)F(B)$, si es norma sub-multiplicativa

Sean f_1, f_2 normas vectoriales, se define $F(A) : \mathbb{R}^{m \times n} \rightarrow \mathbb{R}$ norma matricial inducida si:

$$F(A) = \max_{x \neq 0} \frac{f_1(Ax)}{f_2(x)} = \max_{x: f_2(x)=1} f_1(Ax)$$

Por ejemplo se define $\|\cdot\|_2$ matricial como:

$$\|A\|_2 = \max_{x: \|x\|_2=1} \|Ax\|_2$$

Y así con cualquier norma vectorial.

- $\|A\|_1 = \max_j (\|c_j(A)\|_1)$
- $\|A\|_\infty = \max_i (\|f_i(A)\|_1)$

Sea A inversible y $\|\cdot\|$ norma matricial, se define número de condición de A como:

$$\kappa(A) = \|A\| \|A^{-1}\|$$

- Si $\|\cdot\|$ es norma inducida $\implies \kappa(I) = 1$
- Si $\|\cdot\|$ es norma sub-multiplicativa $\implies \kappa(A) \geq 1$

Intuición: Da una idea de qué tan bien o mal condicionado está el sistema de ecuaciones.

$\kappa(A)$ grande \implies pequeños cambios en el vector imagen (b) **pueden** causar grandes cambios en el vector origen (x).

Sea \hat{x} solución aproximada de $Ax = b$, $b \neq 0$ y $r = Ax - A\hat{x} = b - \hat{b}$:

$$\frac{\|x - \hat{x}\|}{\|x\|} \leq \|A\| \|A^{-1}\| \frac{\|b - \hat{b}\|}{\|b\|}$$

Entonces, si $\kappa(A)$ podemos asegurar que pequeños cambios en b resultan en pequeños cambios en la solución x .

1.3 Matrices SDP

A es una matriz *sdp* (*Simétrica Definida Positiva*) \iff 1. A es simétrica: $A = A^t$ 2. A es definida positiva: $x^t A x > 0 \forall x \neq 0$

Equivalencias:

- A *sdp* $\iff A$ tiene factorización Cholesky ($A = LL^t$, L triangular inferior)
- A *sdp* $\iff A$ tiene todas sus submatrices principales con determinante positivo

Propiedades:

- A es inversible

- $a_{ii} > 0 \forall i$
- Toda submatriz principal es sdp
- Tiene factorización LU y se puede hacer EG sin intercambiar filas
- La submatriz $[2 : n, 2 : n]$ luego del primer paso de EG es sdp
- $A \text{ } sdp \iff B^t A B \text{ } sdp$ con B inversible

Factorización de Cholesky: $A = LL^t$ con L triangular inferior.

Utilizando que A es sdp , podemos construir esta factorización (sabiendo que A tiene factorización LU):

$$A = LU \iff A = LDL^t \text{ (utilizando simetría)} \iff A = L\sqrt{D}\sqrt{D}L^t \text{ (utilizando definida positiva y } d_{ii} > 0) \iff A = (L\sqrt{D})(L\sqrt{D})^t = \hat{L}\hat{L}^t$$

El algoritmo para obtener la factorización cholesky tiene complejidad $\mathcal{O}(n^3)$

1.4 Factorización QR

Motivación: Resolver sistemas de ecuaciones más eficientemente.

$$Q \in \mathbb{R}^{n \times n} \text{ es ortogonal} \iff QQ^t = Q^tQ = I$$

Propiedades:

- Columnas/filas ortonormales
- $\|Q\|_2 = 1$
- $\kappa_2(Q) = 1$, es una matriz estable
- $\|Qx\|_2 = \|x\|_2$, preserva la norma 2
- Producto de ortogonales es ortogonal
- $|\det(Q)| = 1$

Si conocemos Q ortogonal R triangular superior, tal que $A = QR$, entonces:

$$Ax = b \iff Rx = Q^tb$$

Y podemos resolver ese sistema en $\mathcal{O}(n^2)$

Algoritmos para encontrar factorización QR :

- **Método de givens (rotaciones)**

Utilizando la matriz ortogonal que rota a todo vector del plano en ángulo θ en sentido horario

$$W = \begin{pmatrix} \cos(\theta) & \sin(\theta) \\ -\sin(\theta) & \cos(\theta) \end{pmatrix}$$

Buscamos la W tal que dados $\hat{x}, \hat{y} = \begin{pmatrix} \|\hat{x}\|_2 \\ 0 \end{pmatrix}$ se cumpla $W\hat{x} = \hat{y}$

Por cada elemento de la matriz A que este debajo de la diagonal, usamos una W_{ij} que ponga un 0, “metiendo” la W es una matriz identidad.

$$W_{n-1,n}W_{n-2,n} \cdots W_{1,n} \cdots W_{1,2}A = R$$

$$A = (W_{1,2}^t \cdots W_{1,n}^t \cdots W_{n-2,n}^t W_{n-1,n})R$$

$$A = QR$$

Costo total del algoritmo: $\mathcal{O}(n^3)$

- **Método de Householder (reflexiones)**

Utilizando la matriz que refleja a todo vector respecto a un plano dado por u, v . H ortogonal tal que:

$$H\hat{x} = \hat{y}$$

$$Hu = -u$$

$$Hv = v$$

u, v forman una base, entonces $\hat{x} = \alpha u + \beta v$ y $\hat{y} = \alpha u - \beta v$

La matriz H que buscamos es $H = I - 2uu^t$

- H es simétrica y ortogonal

Sean $\hat{x}, \hat{y}, \hat{x} \neq \hat{y}, \|\hat{x}\|_2 = \|\hat{y}\|_2$ existe una H tal que $H\hat{x} = \hat{y}$

Llamamos $v = \hat{x} + \hat{y}$ y $u = \frac{\hat{x} - \hat{y}}{\|\hat{x} - \hat{y}\|_2}$

Ahora en cada paso vamos a tomar \hat{x} como una columna de A (desde la diagonal) e

$$\hat{y} = \begin{pmatrix} \|\hat{x}\|_2 \\ 0 \\ \vdots \\ 0 \end{pmatrix}$$

De nuevo “metiendo” la H en una identidad, ya que en cada paso es de una dimensión menor.

$$H_{n-1}H_{n-2} \cdots H_1 A = R$$

$$A = (H_1^t \cdots H_{n-1}^t)R$$

$$A = QR$$

Costo total del algoritmo: $\mathcal{O}(n^3)$

Propiedades:

- Sea A no singular, existen únicas Q ortogonal y R triangular superior con $r_{ii} > 0$ tal que $A = QR$
- Toda matriz tiene factorización QR

1.5 Autovalores

Sea $A \in \mathbb{R}^{n \times n}$ se dice $\lambda \in \mathbb{R}$ *autovalor* si existe $v \in \mathbb{R}^n$, $v \neq 0$ tal que

$$Av = \lambda v$$

$$\text{Radio espectral: } \rho(A) = \max_{\lambda \text{ autovalor}} |\lambda|$$

- $A - \lambda I$ es no inversible
- Polinomio característico: $P(\lambda) = \det(A - \lambda I)$ se cumple que: λ autovalor $\iff \lambda$ raíz de $P(\lambda)$
- A tiene n autovalores contados con multiplicidad
- v autovector $\implies \alpha v$ autovector

Propiedades:

- λ autovalor de $A \implies \lambda - \alpha$ autovalor de $A - \alpha I$
- λ autovalor de A con v asociado $\implies \lambda^k$ autovalor de A^k con v asociado
- Q ortogonal \implies autovalores reales son 1 o -1
- Si $\lambda_1, \lambda_2, \dots, \lambda_k$ autovalores distintos $\implies v_1, v_2, \dots, v_k$ son L.I.
- A y A^t tienen mismos autovalores

Discos de Gershgorin Sea $A \in \mathbb{C}^{n \times n}$ y $r_i = \sum_{k=1, k \neq i}^n |a_{ik}|$

$$D_i = \{x \in \mathbb{C} : |x - a_{ii}| \leq r_i\}$$

- Si λ autovalor de $A \implies \lambda \in D_i$ para algún i - Si $M = D_{i_1} \cup D_{i_2} \cup \dots \cup D_{i_m}$ es disjunto del resto de los $D_i \implies$ hay exactamente m autovalores en M contados con multiplicidad.

Definición: A, B son *semejantes* si existe P inversible tal que:

$$A = P^{-1}BP$$

- A y B tienen los mismos autovalores

A es *diagonalizable* si es semejante a una matriz diagonal:

$$A \text{ diagonalizable} \iff \text{tiene base de autovectores}$$

Sea $A \in \mathbb{R}^{n \times n}$ vale:

- Si A simétrica \implies tiene autovalores reales
- Si A tiene autovalor real \implies existe autovector asociado real
- A simétrica y λ_1, λ_2 distintos $\implies v_1 \perp v_2$
- Q ortogonal, λ autovalor de $A \iff \lambda$ autovalor de $Q^t A Q$ (semejantes)
- Si A tiene autovalores reales $\implies \exists Q$ ortogonal tal que $Q^t A Q = T$ con T triangular superior
- Si A es simétrica $\implies T$ es diagonal y elementos de T son autovalores de A y columnas de Q autovectores
- Si A es simétrica \implies tiene base de autovectores

Esto último implica que si A es simétrica, entonces es diagonalizable.

Método de la potencia Sea $A \in \mathbb{R}^{n \times n}$, $|\lambda_1| > |\lambda_2| \geq \dots \geq |\lambda_n|$ y $q^0, \|q^0\|_2 = 1$. Se define la siguiente sucesión:

$$q^k = \frac{Aq^{k-1}}{\|Aq^{k-1}\|_2}$$

Si $q_0 = \sum_{i=1}^n \alpha_i v_i$ con $\alpha_1 \neq 0$ (utiliza v_1 en la C.L.), esta sucesión converge al autovector v_1 , y $\lambda_k = (q^k)^t A q^k$ converge a λ_1 .

Este método nos da el autovalor y autovector dominante. Se lo puede combinar con **deflación** para obtener los siguientes autovalores:

Sea H matriz ortogonal tal que $Hv_1 = e_1$ (asumiendo $\|v_1\|_2 = 1$):

$$HAH^t = \begin{pmatrix} \lambda_1 & a^t \\ 0 & B \end{pmatrix}$$

Como A y HAH^t tienen los mismos autovalores, los otros autovalores de A corresponden a la matriz B ($\lambda_2, \dots, \lambda_n$).

1.6 Factorización SVD

Motivación: Tener una factorización que sirva para cualquier matriz para resolver diferentes problemas. Por ej: para resolver cuadrados mínimos. No se necesita que la matriz sea cuadrada.

Vamos a escribir a una matriz $A \in \mathbb{R}^{m \times n}$ como $A = U\Sigma V^t$ donde $U \in \mathbb{R}^{m \times m}$, $V \in \mathbb{R}^{n \times n}$ son ortogonales y $\Sigma \in \mathbb{R}^{m \times n}$ es “diagonal”.

Sea $r = \text{rg}(A)$, la matriz Σ tiene r valores $\sigma_i > 0$ en la diagonal, ordenados de mayor a menor. Y luego completa con 0.

$$\sigma_1 \geq \sigma_2 \geq \dots \geq \sigma_r > 0$$

Estos son los *valores singulares* de la matriz A . Se cumple

- v_1, \dots, v_n autovectores de $A^t A$ son las columnas de V
- u_1, \dots, u_m autovectores de AA^t son las columnas de U
- $\sigma_i = \sqrt{\lambda_i}$ con λ_i iésimo autovalor de $A^t A$ ($\lambda_1 \geq \lambda_2 \geq \dots \geq \lambda_r$)

Entonces también debe valer:

- $Av_i = \sigma_i u_i$ con $i = 1, \dots, r$
- $Av_i = 0$ con $i = r + 1, \dots, n$
- $A^t u_i = \sigma_i v_i$ con $i = 1, \dots, r$
- $A^t u_i = 0$ con $i = r + 1, \dots, m$

AA^t y $A^t A$ son matrices simétricas y semi-definidas positivas, por lo que sus autovalores son mayores o iguales a 0. Tienen exactamente r autovalores positivos.

Propiedades:

- $\|A\|_2 = \sigma_1$
- $\kappa_2(A) = \frac{\sigma_1}{\sigma_n}$
- $\|A\|_F = \sqrt{\sigma_1^2 + \sigma_2^2 + \dots + \sigma_r^2}$

Para conseguirla: Calcular AA^t (ó A^tA), calcular sus autovalores y autovectores con eso tendremos U (ó V) y los valores singulares. Luego armar V (ó U) utilizando las ecuaciones.

1.7 Métodos Iterativos

Motivación: armar una sucesión que converge a la solución del sistema con el objetivo de que sea más eficiente que los métodos exactos (como EG).

Esquema básico: dado $x^0 \in \mathbb{R}^n$ se define $x^{k+1} = Tx^k + C$.

A es una matriz convergente (converge $\forall x^0$ inicial) \iff 1. $\lim_{x \rightarrow \infty} A_{ij}^k = 0$ 2. $\rho(A) < 1$ 3. $\lim_{x \rightarrow \infty} \|A^k\| = 0$ para toda norma inducida 4. $\lim_{x \rightarrow \infty} A^k x = 0 \forall x \in \mathbb{R}^n$

Propiedad: - Si $\rho(A) < 1 \implies I - A$ es no singular y $\sum_{k=0}^{\infty} A^k = (I - A)^{-1}$

Vamos a separar $A = D - L - U$, D tiene los elementos de la diagonal, L, U tienen los elementos de arriba y abajo de la diagonal cambiados de signo.

$$Ax = b \iff (D - L - U)x = b$$

• Método de Jacobi

$$x_i^{k+1} = (b_i - \sum_{j=1, j \neq i}^n a_{ij}x_j^k)/a_{ii}$$

Que se puede escribir matricialmente de la siguiente manera:

$$x^{k+1} = D^{-1}(L + U)x^k + D^{-1}b$$

Asumiendo $a_{ii} \neq 0 \forall i$, ya que D tiene que ser inversible. Utilizando la ecuación anterior, se puede ver que si converge lo hace para la solución del sistema.

• Método de Gauss-Seidel

$$x_i^{k+1} = (b_i - \sum_{j=1}^{i-1} a_{ij}x_j^{k+1} - \sum_{j=i+1}^n a_{ij}x_j^k)/a_{ii}$$

La idea sería aprovechar los coeficientes ya calculados (los x_j^{k+1}), intuyendo que puede ayudar a la convergencia.

$$x^{k+1} = (D - L)^{-1}Ux^k + (D - L)^{-1}b$$

Asumiendo $a_{ii} \neq 0 \forall i$, ya que $D - L$ tiene que ser inversible.

Matrices particulares: - Si A es *edd* \implies método de *Jacobi* converge - Si A es *edd* \implies método de *Gauss-Seidel* converge - Si A es *sdp* \implies método de *Gauss-Seidel* converge

Cota del error: Sea T tal que $\|T\| < 1$ para alguna norma inducida: - T es convergente: $\rho(T) \leq \|T\|$ para toda norma inducida. - $\|x - x^k\| \leq \|T\|^k \|x^0 - x\|$ - $\|x - x^k\| \leq \frac{\|T\|^k}{1 - \|T\|} \|x^1 - x^0\|$

1.8 Cuadrados Mínimos

Motivación: encontrar una función que aproxime a un conjunto de datos.

Tenemos (x_i, y_i) con $i = 1, \dots, m$. Vamos a buscar $f \in \mathcal{F}$ tal que:

$$\min_{f \in \mathcal{F}} (\max_{1 \leq i \leq m} |f(x_i) - y_i|) \text{ (minimax)}$$

$$\min_{f \in \mathcal{F}} (\sum_{i=1}^m |f(x_i) - y_i|)$$

$$\min_{f \in \mathcal{F}} (\sum_{i=1}^m (f(x_i) - y_i)^2) \text{ (cuadrados mínimos)}$$

El problema de *minimax* es que un dato “malo” puede afectar mucho. Se utiliza cuadrados mínimos por derivabilidad.

Dado un conjunto de funciones $\{\phi_1, \dots, \phi_n\}$ LI $\implies \mathcal{F} = \{f(x) = \sum_{j=1}^n c_j \phi_j(x)\}$ El problema de CML se puede escribir:

$$\min_{f \in \mathcal{F}} (\sum_{i=1}^n (f(x_i) - y_i)^2) = \min_{c_1, \dots, c_n} (\sum_{i=1}^n (\sum_{j=1}^n c_j \phi_j(x_i) - y_i)^2)$$

Y esto lo podemos escribir matricialmente de la siguiente manera, $A \in \mathbb{R}^{m \times n}, b \in \mathbb{R}^m, x \in \mathbb{R}^n$:

$$A = \begin{pmatrix} \phi_1(x_1) & \phi_2(x_1) & \cdots & \phi_n(x_1) \\ \phi_1(x_2) & \phi_2(x_2) & \cdots & \phi_n(x_2) \\ \vdots & \vdots & \cdots & \vdots \\ \phi_1(x_m) & \phi_2(x_m) & \cdots & \phi_n(x_m) \end{pmatrix} b = \begin{pmatrix} y_1 \\ y_2 \\ \vdots \\ y_m \end{pmatrix} x = \begin{pmatrix} c_1 \\ c_2 \\ \vdots \\ c_n \end{pmatrix}$$

Y el problema de CML se formula como:

$$\min_{x \in \mathbb{R}^n} \|Ax - b\|_2^2$$

Utilizando la siguiente propiedad: $Im(A) \oplus Nu(A^t) = \mathbb{R}^n$, podemos escribir $b = b_1 + b_2$ con $b_1 \in Im(A), b_2 \in Nu(A^t)$ Y podemos reescribir CML:

$$\min_{x \in \mathbb{R}^n} \|Ax - b\|_2^2 = \min_{y \in Im(A)} \|y - b_1\|_2^2 + \|b_2\|_2^2 \text{ (utilizando } b_1 \perp b_2)$$

El segundo término no depende de y , entonces la solución al problema es $y = b_1$, que siempre existe porque $b_1 \in Im(A)$. - La solución es única \iff columnas de A son LI

Ecuaciones normales: Para hallar la solución de cuadrados mínimos alcanza con resolver:

$$A^t A x = A^t b$$

Y si $rg(A) = n \implies A^t A$ inversible \implies

$$x = (A^t A)^{-1} A^t b$$

Maneras de resolver el problema de CML: - Utilizando QR Vamos a separar en 2 casos:

1. $rg(A) = n$ En este caso, $A = Q \begin{pmatrix} R \\ 0 \end{pmatrix}$ y $r_{ii} \neq 0$ ya que $rg(A) = n$, entonces R inversible.

$$\min_{x \in \mathbb{R}^n} \|Ax - b\|_2^2 = \min_{x \in \mathbb{R}^n} \|Q^t Ax - Q^t b\|_2^2 = \min_{x \in \mathbb{R}^n} \|Rx - b^{(n)}\|_2^2 + \|b^{(m-n)}\|_2^2$$

Llamamos $Q^t b = (b^{(n)}, b^{(m-n)})$, por lo tanto la solución es: $x = R^{-1}b^{(n)}$

2. $rg(A) = r < n$
En este caso, $\hat{A} = Q \begin{pmatrix} R_1 & R_2 \\ 0 & 0 \end{pmatrix}$, $R_1 \in \mathbb{R}^{r \times r}$ triangular superior de rango r y $\hat{A} = AP$ una permutación de la matriz para que queden elementos no nulos en los primeros r lugares.

Llamamos $P\hat{x} = x$:

$$\min_{x \in \mathbb{R}^n} \|Ax - b\|_2^2 = \min_{\hat{x} \in \mathbb{R}^n} \|AP\hat{x} - b\|_2^2 = \min_{\hat{x} \in \mathbb{R}^n} \|Q^t AP\hat{x} - Q^t b\|_2^2$$

$$\min_{\hat{x} \in \mathbb{R}^n} \|R_1 \hat{x}^{(r)} + R_2 \hat{x}^{(n-r)} - b^{(r)}\|_2^2 + \|b^{(m-r)}\|_2^2$$

$Q^t b = (b^{(r)}, b^{(m-r)})$, por lo tanto la solución es: $R_1 \hat{x}^{(r)} = b^{(r)} - R_2 \hat{x}^{(m-r)}$ Notar que hay que volver a obtener $x = P\hat{x}$, y que tenemos libertad para $R_2 \hat{x}^{(m-r)}$

- Utilizando SVD Utilizando $A = U\Sigma V^t \iff U^t A = \Sigma V^t$

$$\min_{x \in \mathbb{R}^n} \|Ax - b\|_2^2 = \min_{x \in \mathbb{R}^n} \|U^t Ax - U^t b\|_2^2 = \min_{x \in \mathbb{R}^n} \|\Sigma V^t x - U^t b\|_2^2$$

Sea $y = V^t x$:

$$\min_{y \in \mathbb{R}^n} \|\Sigma y - U^t b\|_2^2 = \min_{y \in \mathbb{R}^n} \sum_{i=1}^r (\sigma_i y_i - (U^t b)_i)^2 + \sum_{i=r+1}^m ((U^t b)_i)^2$$

La solución a esto será:

$$y_i = \frac{(U^t b)_i}{\sigma_i} \quad \forall i = 1, \dots, r$$

Y puede tener cualquier valor para $i = r+1, \dots, n$. Luego la solución a CML es $x = Vy$

- Si se busca x de mínima norma 2 $\implies y_i = 0 \quad \forall i = r+1, \dots, n$

1.9 Interpolación

Motivación: encontrar una función que pase *exactamente* por un conjunto de puntos.

Al igual que en CML, vamos a tener un conjunto de puntos (x_i, y_i) , pero buscamos una función (en particular un polinomio) que los *interpole*, es decir que pase exactamente por esos puntos.

Buscamos $P(x)$ polinomio de grado $\leq n$ tal que $P(x_i) = y_i \quad \forall i$.

Vamos a utilizar los siguientes polinomios L :

$$L_{nk} = \prod_{i=0, i \neq k}^n \frac{x - x_i}{x_k - x_i}$$

Este polinomio cumple:

- Es de grado n
- $L_{nk}(x_i) = 0 \quad \forall i \neq k$
- $L_{nk}(x_k) = 1$

Con esto construimos el polinomio que queremos de la siguiente manera:

$$P(x) = \sum_{k=0}^n y_k L_{nk}$$

- $P(x)$ es de grado $\leq n$
- $P(x_i) = y_i \quad \forall i$
- Este polinomio es único

Error: Sea $f(x) \in C^{n+1}[a, b]$, $P(x)$ polinomio interpolante, $\hat{x} \in [a, b]$

$$\exists \xi(\hat{x}) : f(\hat{x}) = P(\hat{x}) + \frac{f^{n+1}(\xi(\hat{x}))}{(n+1)!} (\hat{x} - x_0)(\hat{x} - x_1) \cdots (\hat{x} - x_n)$$

Utilizando esta fórmula del error, se puede demostrar la unicidad del polinomio interpolante. Asumiendo que hay P_1, P_2 polinomios interpolantes y viendo que $P_1 = P_2$.

Otras maneras de construir el polinomio:

- **Diferencias divididas** Dados $(x_i, f(x_i))$:
 1. **Orden 0:** $f[x_i] = f(x_i)$
 2. **Orden 1:** $f[x_i, x_{i+1}] = \frac{f[x_{i+1}] - f[x_i]}{x_{i+1} - x_i}$
 3. **Orden k:** $f[x_i, x_{i+1}, \dots, x_{i+k}] = \frac{f[x_{i+1}, \dots, x_{i+k}] - f[x_i, \dots, x_{i+k-1}]}{x_{i+k} - x_i}$

Con estas definiciones podemos construir el polinomio de la siguiente manera:

$$P(x) = f[x_0] + f[x_0, x_1](x - x_0) + f[x_0, x_1, x_2](x - x_0)(x - x_1) + \cdots \\ \cdots + f[x_0, \dots, x_k](x - x_0) \cdots (x - x_{k-1}) + f[x_0, \dots, x_n](x - x_0) \cdots (x - x_{n-1})$$

La ventaja de esta escritura es que podemos agregar puntos sin tener que calcular todo el polinomio de cero.

- **Polinomios Q_{ij} (de Neville)**

Dados x_0, \dots, x_k el polinomio interpolante $P_{01\dots k}$ puede expresarse:

$$P_{01\dots k} = \frac{(x - x_j)P_{01\dots j-1j+1\dots k} - (x - x_i)P_{01\dots i-1i+1\dots k}}{x_i - x_j}$$

Sea Q_{ij} polinomio interpolador de grado $\leq j$ en los puntos $x_{i-j}, x_{i-j+1}, \dots, x_i$

$$Q_{ij} = P_{i-j\dots i} \\ Q_{ij} = \frac{(x - x_{i-j})Q_{ij-1} - (x - x_i)Q_{i-1j-1}}{x_i - x_{i-j}}$$

De esta manera representamos al polinomio en función de dos polinomios que interpolan un punto menos, y al momento de agregarse un punto nuevo podemos construirlo, ya que el polinomio se construye de manera incremental.

$$P(x) = Q_{nn}(x) = \frac{(x - x_0)Q_{nn-1}(x) - (x - x_n)Q_{n-1n-1}(x)}{x_n - x_0}$$

Esto permite establecer una recursión $\Rightarrow Q_{nn}$ depende de Q_{n-1n-1} y Q_{nn-1}

Interpolación segmentaria:

La idea es no interpolar todos los puntos con una función, sino tomar de a 2 puntos e interpolarlos intentando evitar polinomios de grado alto que pueden tener muchas oscilaciones.

Sean (x_i, y_i) para $i = 0, \dots, n$ con $x_i < x_{i+1}$: 1. **Lineal**: interpolación lineal entre cada par de puntos: Tenemos n polinomios de la pinta $L_i(x) = a_i + b_i(x - x_i)$: - 2 incógnitas por cada uno $\Rightarrow 2n$ incógnitas - 2 ecuaciones por cada uno (para interpolar) $\Rightarrow 2n$ ecuaciones - $L_i(x_i) = y_i$ - $L_i(x_{i+1}) = y_{i+1}$ Cada L_i queda unívocamente determinado. El problema es que nos da una función continua pero no derivable, por eso muchas veces es deseado utilizar un grado mayor.

2. **Cuadrática**: interpolación cuadrática entre cada par de puntos Tenemos n polinomios $Q_i(x) = a_i + b_i(x - x_i) + c_i(x - x_i)^2$

- 3 incógnitas por cada uno $\Rightarrow 3n$ incógnitas
- 2 ecuaciones por cada uno $\Rightarrow 2n$ ecuaciones
 - $Q_i(x_i) = y_i$
 - $Q_i(x_{i+1}) = y_{i+1}$
- Y podemos pedir también: $Q'_i(x_{i+1}) = Q'_{i+1}(x_{i+1}) \Rightarrow n-1$ ecuaciones (*derivabilidad*)

Tenemos $3n$ incógnitas y $3n-1$ ecuaciones. Se le puede agregar una condición en la derivada de alguno de los extremos, pero se pierde simetría.

3. **Cúbica** Tenemos n polinomios $S_i(x) = a_i + b_i(x - x_i) + c_i(x - x_i)^2 + d_i(x - x_i)^3$

- 4 incógnitas por cada uno $\Rightarrow 4n$ incógnitas
- 2 ecuaciones por cada uno $\Rightarrow 2n$ ecuaciones
 - $S_i(x_i) = y_i$
 - $S_i(x_{i+1}) = y_{i+1}$
- Para derivabilidad pedimos:
 - $S'_i(x_{i+1}) = S'_{i+1}(x_{i+1}) \Rightarrow n-1$ ecuaciones
 - $S''_i(x_{i+1}) = S''_{i+1}(x_{i+1}) \Rightarrow n-1$ ecuaciones

Tenemos $4n$ incógnitas y $4n-2$ ecuaciones. Hay dos alternativas para las ecuaciones restantes: - Que la derivada segunda en los extremos se anule: $S''_0(x_0) = S''_{n-1}(x_n) = 0$ - Que la derivada primera en los extremos coincida con la del polinomio interpolante: $S'_0(x_0) = f'(x_0)$ y $S'_{n-1}(x_n) = f'(x_n)$

En ambos casos el sistema es *edd*, por lo que la solución siempre existe y es única. Esto se demuestra dando la solución unívoca del sistema.

1.10 Aritmética de la computadora

Motivación: analizar la manera en la que se representan los números reales en la computadora.

En una computadora solo podemos representar números con una cantidad de dígitos fija y finita, por lo que se diferencian de los números reales. Esto genera que al hacer cálculos en una computadora, esta puede generar *error numérico*.

- **Float:** estándar de la *IEEE* para representar números con decimales. Utiliza signo, mantisa y exponente.

Se dice *underflow* cuando un resultado es menor que el mínimo representable, y *overflow* cuando sucede con el máximo. Depende de la cantidad de bits utilizados (*single*, *double*, etc.)

Los números representados no están uniformemente distribuidos sobre la recta real, cuanto más nos alejamos del 0, más espacio habrá entre dos números.

Sea x^* el valor que pretende aproximar a x :

- **Error real:** $x - x^*$
- **Error absoluto:** $|x - x^*|$
- **Error relativo:** $\frac{|x - x^*|}{|x|}$

Esto se extiende a vectores cuando se trabaja en \mathbb{R}^n

Se puede definir un ϵ en la máquina tal que: $\epsilon = \frac{1}{2}\beta^{1-t}$, donde β es la base y t es la precisión
- $\implies \forall x \exists x^*$ tal que $\frac{|x - x^*|}{|x|} \leq \epsilon$ Es decir ϵ es una cota para el error relativo.

Problemas en los cálculos:

1. **Cancelación:** Al restar dos números cercanos entre sí, el resultado será cercano al 0 por lo que se pueden perder dígitos significativos. Esto se puede esperar cuando $|a - b| \ll |a| + |b|$
2. **Producto con números grandes/División con números pequeños:** En este caso se observa una amplificación del error cometido. Al dividir a x^* por 10^{-n} , el error absoluto cometido se multiplica por un factor 10^n .
3. **Suma:** Al sumar muchos términos, el error cometido total va a variar dependiendo del orden de los sumandos. Para esto hay dos propuestas: ordenar ascendentemente o descendientemente los números antes de sumar. También existe el algoritmo *suma Kahan*, que intenta minimizar el error cometido al sumar muchos números.